SCE Training Curriculum
for Integrated Automation Solutions
Totally Integrated Automation (TIA)

Siemens Automation Cooperates with Education

TIA Portal Module 010-070
Communication between two SIMATIC S7-1200
Matching SCE training packages for these training curriculums

- **SIMATIC S7-1200 AC/DC/RELAY 6er "TIA Portal"**
  Order number: 6ES7214-1BE30-4AB3

- **SIMATIC S7-1200 DC/DC/DC 6er "TIA Portal"**
  Order number 6ES7214-1AE30-4AB3

- **SIMATIC S7-SW for Training STEP 7 BASIC V11 Upgrade (for S7-1200) 6er "TIA Portal"**
  Order number 6ES7822-0AA01-4YE0

Please note that these training packages are replaced with successor packages when necessary. An overview of the currently available SCE packages is provided under: siemens.com/sce/tp

**Continued Training**
For regional Siemens SCE continued training, please contact your regional SCE contact person siemens.com/sce/contact

**Additional information regarding SCE**
siemens.com/sce

**Information regarding Usage**

This SCE training curriculum for the integrated automation solution Totally Integrated Automation (TIA) was prepared for the program "Siemens Automation Cooperates with Education (SCE)" specifically for training purposes for public education facilities and R&D facilities. Siemens AG does not guarantee the contents.

This document is to be used only for initial training on Siemens products/systems; i.e., it can be copied entirely or partially and given to those being trained for usage within the scope of their training. Passing on as well as copying this training curriculum and sharing its content is permitted within public training and advanced training facilities for training purposes.

Exceptions require written permission by the Siemens AG contact person: Roland Scheuerer roland.scheuerer@siemens.com.

Offenders will be held liable. All rights including translation are reserved, particularly if a patent is granted or a utility model or design is registered.

Usage for industrial customer courses is explicitly not permitted. We do not consent to the training curriculums being used commercially.

We wish to thank the Michael Dziallas Engineering Corporation and all other involved persons for their support during the preparation of this training curriculum.
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Preface</td>
<td>4</td>
</tr>
<tr>
<td>2. Instructions on Programming the SIMATIC S7-1200</td>
<td>6</td>
</tr>
<tr>
<td>2.1 Automation System SIMATIC S7-1200</td>
<td>6</td>
</tr>
<tr>
<td>2.2 Programming Software STEP 7 Professional V11 (TIA Portal V11)</td>
<td>6</td>
</tr>
<tr>
<td>2.3 SIMATIC NET Switch CSM 1277</td>
<td>7</td>
</tr>
<tr>
<td>3. Conveyor Control with Counter and Multi-Instance</td>
<td>8</td>
</tr>
<tr>
<td>3.1 Task</td>
<td>8</td>
</tr>
<tr>
<td>4. Expanding the Hardware Configuration in the Project Conveyor Control</td>
<td>9</td>
</tr>
<tr>
<td>4.1 Loading the Sample Project and Re-Storing It</td>
<td>9</td>
</tr>
<tr>
<td>4.2 Adding the Second CPU</td>
<td>13</td>
</tr>
<tr>
<td>4.3 Connecting the Controllers</td>
<td>15</td>
</tr>
<tr>
<td>4.4 Loading the Hardware Configuration to the CPUs</td>
<td>16</td>
</tr>
<tr>
<td>5. Programming the Program Blocks</td>
<td>21</td>
</tr>
<tr>
<td>5.1 Send Block TSEND_C</td>
<td>21</td>
</tr>
<tr>
<td>5.2 Control Program for the Controller_Conveyor</td>
<td>24</td>
</tr>
<tr>
<td>5.3 Receive Block TRCV_C</td>
<td>30</td>
</tr>
<tr>
<td>5.4 Control Program for controller_data</td>
<td>35</td>
</tr>
</tbody>
</table>
1. Preface

Regarding its content, module 010-070 is part of the training unit 'Basics of PLC Programming' and explains the communication between two SIMATIC S7-1200 controllers.

Training Objective

In this module 010-070, the reader learns how to configure an ISO on TCP communication connection between two SIMATIC S7-1200 controllers, and how to program the data exchange in the control program using the communication blocks TSEND_C and TRCV_C.

Prerequisites

To successfully work through this module 010-070, the following knowledge is assumed:
- How to handle Windows
- Basics of PLC programming with the TIA Portal
  (for example, Module 010-010 – 'Startup' programming of the SIMATIC S7-1200 with TIA-Portal V11)
- Blocks for the SIMATIC S7-1200
  (for example, Module 010-020 – block types for the SIMATIC S7-1200)
- Timer and counter blocks for the SIMATIC S7-1200
  (for example, Module 010-030 – Multi-instances at the SIMATIC S7-1200)
Hardware and software needed

1. PC Pentium 4, 1.7 GHz, 1 (XP) – 2 (Vista) GB RAM, free disk storage approx. 2 GB
   Operating system Windows XP (Home SP3, Professional SP3)/Windows Vista (Home Premium SP1, Business SP1, Ultimate SP1)
2. Software STEP7 Basic V11 SP2 (Totally Integrated Automation (TIA) Portal V11)
3. Ethernet connection between PC and CPU 1214C and panel
4. PLC 1 SIMATIC S7-1200; for example, CPU 1214C with Switch CSM 1277.
   The inputs have to be brought out to a panel.
5. PLC 2 SIMATIC S7-1200; for example, CPU 1214C with Switch CSM 1277.
   The inputs have to be brought out to a panel.
2. Instructions on Programming the SIMATIC S7-1200

2.1 Automation System SIMATIC S7-1200

The automation system SIMATIC S7-1200 is a modular mini-controller system for the lower and medium performance range.

An extensive module spectrum is available for optimum adaptation to the automation task.

The S7 controller consists of a power supply, a CPU and input/output modules for digital and analog signals.

If needed, communication processors and function modules are added for special tasks such as step motor control.

With the S7 program, the programmable logic controller (PLC) monitors and controls a machine or a process, whereby the IO modules are polled in the S7 program by means of the input addresses (%I) and addressed by means of output addresses (%Q).

The system is programmed with the software STEP 7.

2.2 Programming Software STEP 7 Professional V11 (TIA Portal V11)

The software STEP 7 Professional V11 (TIA Portal V11) is the programming tool for the following automation systems

- SIMATIC S7-1200
- SIMATIC S7-300
- SIMATIC S7-400
- SIMATIC WinAC

With STEP 7 Professional V11, the following functions can be utilized to automate a plant:

- Configuring and parameterizing the hardware
- Defining the communication
- Programming
- Testing, commissioning and service with the operating/diagnostic functions
- Documentation
- Generating the visual displays for the SIMATIC basic panels with the integrated WinCC Basic
- With additional WinCC packages, visual display solutions for PCs and other panels can be prepared also.

All functions are supported with detailed online help.
2.3 SIMATIC NET Switch CSM 1277

System Description

The CSM 1277 is equipped with four RJ45 socket connectors to connect terminal equipment or additional network segments.

The TP interfaces are laid out as RJ45 socket connector with MDI-X assignment (Medium Dependent Interface Autocrossover) of a network component. If the receive cable pair is connected wrong (RD+ and RD- reversed), the polarity is adjusted automatically. The MDI/MDIX autocrossover function offers the advantage of integrated cabling without external crossed Ethernet cables being needed. This prevents malfunctions if send and receive lines should be inadvertently switched.

This considerably simplifies the installation for the user.

The CSM 1277 is a plug&play device that does not require setting to be started.

CSM 1277 Displays

Power display 'L' (green LED).

The status of voltage supply is signaled by means of a green LED.

<table>
<thead>
<tr>
<th>State</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LED is green</td>
<td>Voltage supply is connected</td>
</tr>
<tr>
<td>LED not lit</td>
<td>Voltage supply is not connected, or the applied voltage is too low</td>
</tr>
</tbody>
</table>

Port status display 'P1' to 'P4' (green LED)

The status of the interfaces is signaled by means of the 4 green LEDs. They are located below the upper cover. Refer also to Figure 4-4.

<table>
<thead>
<tr>
<th>State</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Port 1 to Port 4 LED is lit</td>
<td>Existing connection via port to Industrial Ethernet (LINK status)</td>
</tr>
<tr>
<td>Port 1 to Port 4 LED flashes</td>
<td>Port is sending /receiving via Industrial Ethernet</td>
</tr>
<tr>
<td>Port 1 to Port 4 LEDs are flashing/running light</td>
<td>Test phase during power on</td>
</tr>
</tbody>
</table>
3. **Conveyor Control with Counter and Multi-Instance**

Below, the example for conveyor control is to be expanded with a counter and a multi-instance for the communication of two S7-1200 controllers.  

With the conveyor, 20 bottles are always to be transported in a case. When the case is full, the conveyor is stopped, and the case has to be exchanged.

With button 'S1' the operating mode 'Manual' and with button 'S2' the operating mode 'Automatic' can be selected.

In the operating mode 'Manual', the motor is switched on as long as button 'S3' is operated; button 'S4' must not be operated.

In the 'Automatic' mode, the conveyor motor is switched on with button 'S3' and switched off with button 'S4' (NC).

In addition, there is a sensor 'B0' that counts the bottles into the case. After 20 bottles are counted, the conveyor is stopped...

When a new case is put in place, it has to be confirmed with 'S5'.

**Assignment list**

<table>
<thead>
<tr>
<th>Address</th>
<th>Symbol</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>%I 0.0</td>
<td>S1</td>
<td>Button manual mode S1 NO</td>
</tr>
<tr>
<td>%I 0.1</td>
<td>S2</td>
<td>Button automatic mode S2 NO</td>
</tr>
<tr>
<td>%I 0.2</td>
<td>S3</td>
<td>On button S3 NO</td>
</tr>
<tr>
<td>%I 0.3</td>
<td>S4</td>
<td>Off button S4 NC</td>
</tr>
<tr>
<td>%I 0.6</td>
<td>S5</td>
<td>Button S5 NO reset counter/new case</td>
</tr>
<tr>
<td>%I 0.7</td>
<td>B0</td>
<td>Sensor B0 NO bottle counter</td>
</tr>
<tr>
<td>%O 0.2</td>
<td>M01</td>
<td>Conveyor motor M01</td>
</tr>
</tbody>
</table>

3.1 **Task**

The states of the input signals are to be sent to the second CPU for further processing and displayed there.

The inputs of the first CPU are to be written to the Send buffer. With the send block "TSEND_C", the content of the Send buffer is to be sent to the second CPU by means of an ISO on TCP connection.

Inputs of CPU1 from %I0.0 to %I0.7 (%I0B0) to the – %MB10 (send buffer) – "TSEND_C" – CPU2.  

At the second CPU, the received data is to be written to the receive buffer by means of the receive block "TRCV_C". Then the content of the receive buffer is to be displayed at the CPU’s outputs.

CPU2 "TRCV_C" – %MB12 (receive buffer) – to the outputs of %O0.0 to %O0.7 (%OB0).
4. Expanding the Hardware Configuration in the Project Conveyor Control

For project management and programming, the software 'Totally Integrated Automation Portal' is used.
Here, on a uniform interface, components such as controlling, visualizing and networking the automation solution are set up, parameterized and programmed.
Online tools are available for error diagnosis.
In the steps below, the following is to be done: for the SIMATIC S7-1200, a project is to be opened, stored under a different name and adapted to the new requirement:

The central tool is the 'Totally Integrated Automation Portal'. It is called here with a double click.

4.1 Loading the Sample Project and Re-Storing It

The project "FB_conveyor_counter" from Module 010-030 is now opened as a pattern for this program.
Next, 'First steps' are suggested. Click on Open the project view.
Now, the project is to be saved under another name. In the menu **Project** click on "Save As".
Now, 'Save' the project under the new name 'conveyor_CPU_to_data_CPU'. 
4.2 Adding the Second CPU

To set up a second CPU in the project, open the list box with a double click on "Add new device".

Under SIMATIC S7-1200, select CPU 1214C with the order number "6ES7 214-1AE30-0XB0".
As device name, assign "controller_data"
Place a check mark at "Open device view"
Click "OK".
In the lower window of the device view, under the tab **Properties** select the **PROFINET interface**.

Then click on "**Add new subnet**".

Assign the IP address **192.168.0.5** and the subnet mask **255.255.255.0**

Now click on "**Save project**".
4.3 Connecting the Controllers

Change to the Network view.

Drag the line of the PN/IE_1 network to the green square of the controller_conveyor.

The two CPUs are now connected.

Then, click on "Save project".
4.4 Loading the Hardware Configuration to the CPUs

Prior to the steps below, we have to establish the network wiring between the two controllers and the programming device by means of the switch CSM 1277.

In the menu Online, click on "Accessible devices".

The second controller still has the factory settings and no IP address.

Now click on "Cancel".
Loading the first controller conveyor.

In the project navigation window, select "controller_conveyor" and click on the button **Download to device**.

If the CPU is in the RUN mode, it has to be set to STOP prior to the download. Select the PG/PC interface for the download and the connection with subnet.

Now click on "Load".
Prior to downloading, the project data is checked and compiled.

Then click on "Load".

Now click on "Finish".

The CPU returns to the RUN mode.
Downloading Data to Second Controller.

In the project navigation window, select "controller_data" and click on the button **Download to device**.

Select the PG/PC interface for the download and the connection with subnet. Place a checkmark at **Show all accessible devices**. Select the S7-1200 CPU.

Then click on **Load**.
Prior to the download, the project data is checked and compiled.

![Load preview](image)

Then, click on "Load".
Loading writes the IP address to the CPU.

![Load results](image)

Then, click on "Finish".
The CPU is taken to the RUN mode.
5. **Programming the Program Blocks**

Now, the required program blocks for the communication between the two CPUs are to be called and parameterized.

By calling the transmission blocks TSEND_C and TRCV_C, a connection is established automatically between the two CPUs. The connection is set up, established, and monitored automatically until an operation or CPU STOP separates the connection.

5.1 **Send Block TSEND_C**

**Description**

TSEND_C is executed asynchronously and has the following functions:

**Setting up and establishing a communication connection:**

TSEND_C sets up a TCP or ISO-on-TCP communication connection and establishes it. After the connection is set up and established, the CPU maintains and monitors it automatically.

To set up the communication connection, the connection description specified at the parameter CONNECT is used. To establish the connection, the parameter CONT has to be set to "1". If the connection was established successfully, the parameter DONE is set to "1" for 1 cycle.

When the CPU enters the STOP mode, an existing connection is cancelled and the connection that was set up is removed. To again set up and establish the connection, TSEND_C has to be executed again.

The number of possible communication connections is provided in the technical data of your CPU.

**Sending data over an existing communication connection:**

The transmitting range is specified at the parameter DATA. It includes the address and the length of the data to be sent.

The send request is executed when a rising edge is recorded at the parameter REQ. The maximum number of bytes that is sent with a send request is specified at the parameter LEN. The data to be sent must not be edited until the send request is executed completely. If the send request was executed successfully, the parameter DONE is set to "1". The signal status "1" at parameter DONE is not a confirmation, however, that the communication partner has read the data that was sent.

**Disconnecting the communication connection:**

The communication connection is disconnected when the parameter CONT is set to "0".

When setting the parameter COM_RST to "1", TSEND_C is executed again. The existing communication connection is disconnected and a new connection established. Data that is transmitted while the connection is re-established may be lost.
### Parameters of TSEND_C

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Declaration</th>
<th>Data type</th>
<th>Memory area</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>REQ</td>
<td>Input</td>
<td>BOOL</td>
<td>I, Q, M, D, L</td>
<td>Starts the send job on a rising edge.</td>
</tr>
</tbody>
</table>
| CONT      | Input       | BOOL      | I, Q, M, D, L | Controls the communications connection:  
|           |             |           |             | ● 0: Disconnect the communications connection  
|           |             |           |             | ● 1: Establish and maintain the communications connection  
|           |             |           |             | When sending data (rising edge at the REQ parameter), the CONT parameter must have the value "TRUE" in order to establish or maintain a connection. |
| LEN       | Input       | UINT      | I, Q, M, D, L or constant | Maximum number of bytes to be sent with the job. If you use purely symbolic values at the DATA parameter, the LEN parameter must have the value "0". |
| CONNECT   | InOut       | TCON_Param | D          | Pointer to the connection description  
|           |             |           |             | See also: Parameters of communication connections |
| DATA      | InOut       | VARIANT   | I, Q, M, D, L | Pointer to the send area containing the address and the length of the data to be sent. |
| COM_RST   | InOut       | BOOL      | I, Q, M, D, L | Restarts the instruction:  
|           |             |           |             | ● 0: Irrelevant  
|           |             |           |             | ● 1: Complete restart of the instruction causing an existing connection to be terminated and a new connection to be established. |
| DONE      | Output      | BOOL      | I, Q, M, D, L | Status parameter with the following values:  
|           |             |           |             | ● 0: Job not yet started or is still executing  
|           |             |           |             | ● 1: Job completed error-free |
| BUSY      | Output      | BOOL      | I, Q, M, D, L | Status parameter with the following values:  
|           |             |           |             | ● 0: Job not yet started or already completed  
|           |             |           |             | ● 1: Job not yet completed. A new job cannot be started. |
| ERROR     | Output      | BOOL      | I, Q, M, D, L | Status parameter with the following values:  
|           |             |           |             | ● 0: No error  
|           |             |           |             | ● 1: Error occurred |
| STATUS    | Output      | WORD      | I, Q, M, D, L | Status of the instruction |

### BUSY, DONE and ERROR

With the parameters BUSY, DONE, ERROR and STATUS, we can check the execution status. The parameter BUSY shows the processing status.  
The parameter DONE checks whether a request was executed successfully.  
The parameter ERROR is set when errors occur while TSEND_C is executed.

<table>
<thead>
<tr>
<th>BUSY</th>
<th>DONE</th>
<th>ERROR</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-</td>
<td>-</td>
<td>The job is being processed.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>The job was completed successfully.</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>The job was stopped with an error. The cause of the error is specified in the STATUS parameter.</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>No new job was assigned.</td>
</tr>
</tbody>
</table>
The error information is provided at the parameter STATUS.

<table>
<thead>
<tr>
<th>ERROR</th>
<th>STATUS (Var100...)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0000</td>
<td>Job completed error-free</td>
</tr>
<tr>
<td>0</td>
<td>7000</td>
<td>No job processing active</td>
</tr>
<tr>
<td>0</td>
<td>7001</td>
<td>• Start execution of the job</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Establish connection</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Wait for connection partner</td>
</tr>
<tr>
<td>0</td>
<td>7002</td>
<td>Data are being sent</td>
</tr>
<tr>
<td>0</td>
<td>7003</td>
<td>Connection is terminated</td>
</tr>
<tr>
<td>0</td>
<td>7004</td>
<td>Connection established and monitored, no job processing active.</td>
</tr>
<tr>
<td>1</td>
<td>80A0</td>
<td>Group error for error codes 80A1 and 80A2.</td>
</tr>
<tr>
<td>1</td>
<td>80A1</td>
<td>• Connection or port already being used by user.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Communications error:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>o The specified connection has not yet been established.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>o The specified connection is being terminated. A transfer over this connection is not possible.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>o The interface is being re-initialized.</td>
</tr>
<tr>
<td>1</td>
<td>80A2</td>
<td>Local or remote port is being used by the system.</td>
</tr>
<tr>
<td>1</td>
<td>80A3</td>
<td>Attempt being made to terminate a non-existent connection.</td>
</tr>
<tr>
<td>1</td>
<td>80A4</td>
<td>IP address of the remote endpoint of the connection is invalid, in other words, it matches the IP address of the local partner.</td>
</tr>
<tr>
<td>1</td>
<td>80A7</td>
<td>Communications error: You called the COM_RST = 1 instruction before the send job was complete.</td>
</tr>
<tr>
<td>1</td>
<td>80E2</td>
<td>The CONNECT parameter points to a data block that was generated with the attribute Only store in load memory.</td>
</tr>
<tr>
<td>1</td>
<td>80E3</td>
<td>Inconsistent parameter assignment. Group error for error codes 80A0 to 80A2, 80A4, 80A5 to 80A9.</td>
</tr>
<tr>
<td>1</td>
<td>80E4</td>
<td>You have violated one or both of the following conditions for passive connection establishment (active_est = FALSE) when using the ISO-on-TCP protocol variant (connection_type = #16#12): “local_tbaap_id_len &gt;= #16#22”, and/or “local_tbaap_id[1] = #16#20”.</td>
</tr>
<tr>
<td>1</td>
<td>80E5</td>
<td>Only passive connection establishment is permitted for connection type 13 = UDP.</td>
</tr>
<tr>
<td>1</td>
<td>80E6</td>
<td>Parameter assignment error in the connection_type parameter of the data block for connection description.</td>
</tr>
<tr>
<td>1</td>
<td>80E7</td>
<td>Error in one of the following parameters of the data block for connection description: block_length, local_tbaap_id_len, rem_subnet_id_len, rem_staddr_len, rem_tbaap_id_len, next_staddr_len.</td>
</tr>
<tr>
<td>1</td>
<td>80E8</td>
<td>The LEN parameter is higher than the highest permitted value.</td>
</tr>
<tr>
<td>1</td>
<td>80E6</td>
<td>The LEN parameter within the CONNECT parameter is outside the permitted range.</td>
</tr>
<tr>
<td>1</td>
<td>80F7</td>
<td>Maximum number of connections reached; no additional connection possible.</td>
</tr>
<tr>
<td>1</td>
<td>80F8</td>
<td>The value at the LEN parameter does not match the receive area set at the DATA parameter.</td>
</tr>
<tr>
<td>1</td>
<td>80F9</td>
<td>The CONNECT parameter does not point to a data block.</td>
</tr>
<tr>
<td>1</td>
<td>80F1</td>
<td>Maximum nesting depth exceeded.</td>
</tr>
<tr>
<td>1</td>
<td>80F9</td>
<td>The CONNECT parameter points to a field that does not match the length of the connection description.</td>
</tr>
<tr>
<td>1</td>
<td>80FB</td>
<td>The ID of the local device in the connection description does not match the CPU.</td>
</tr>
<tr>
<td>1</td>
<td>80C3</td>
<td>• All connection resources are in use.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• A block with this ID is already being processed in a different priority group.</td>
</tr>
</tbody>
</table>
5.2 Control Program for the Controller_Conveyor

Supplementing the default tag table

Under PLC tags, open the default tag table of the controller_conveyor
Supplement the table.

| 1 | 80C4 | Temporary communications error: |
|   |     | • The connection cannot be established at this time. |
|   |     | • The interface is receiving new parameters or the connection is being established. |
|   |     | • The configured connection is being removed by a "DISCON" instruction. |
|   |     | • The connection used is being terminated by a call with COM_RST=1. |

| 1 | 8722 | CONNECT parameter: The source area is invalid. The area does not exist in the DB. |

| 1 | 879A | CONNECT parameter: Access to the connection description is not possible (for example, DB does not exist). |

| 1 | 877F | CONNECT parameter: Internal error. |

| 1 | 8822 | DATA parameter: Invalid source area, the area does not exist in the DB. |

| 1 | 8834 | DATA parameter: Area error in the VARIANT pointer. |

| 1 | 8832 | DATA parameter: The DB number is too large. |

| 1 | 883A | CONNECT parameter: Access to entered connection data not possible (e.g. because the DB does not exist). |

| 1 | 887F | DATA parameter: Internal error, e.g. illegal VARIANT reference. |

| 1 | 893A | DATA parameter: Access to entered transmission range not possible (e.g. because the DB does not exist). |
The TSEND_C is called in OB1 of the control program.

Open the Main [OB1] of controller_conveyor

Generate the MOVE command in Network 2.
Drag the block "TSEND_C" to Network 3.
In the window that follows, accept the suggested data block. Click on Single instance and then on "OK".

The data block is generated and incorporated automatically. Under Properties, select the connection parameters. At the connection data, select the partner controller and make a selection in the colored fields. For each controller, a new data block has to be selected for the connection, and the connection ID "1" has to be entered directly on both sides.
The send request (REQ) of the block TSEND_C is to take place cyclically—which explains the linking with the output parameter (DONE).

Now, expand the block to its full size and enter the block parameters either at the block or in the property window.

This completes the programming at the controller_conveyor.
Program in function block diagram (FBD):

**Block title:** "Main Program Sweep (Cycle)"

**Comment**

**Network 1:** call program conveyor 1

```
MOVE
"INPUTS" IN  "SENDBUFFER" OUT
```

**Network 2:** copy inputs to send buffer

```
MOVE
"INPUTS" IN  "SENDBUFFER" OUT
```

**Network 3:** call TSEND_C

```
TSEND_C
EN
"DONE"  -  REQ
TRUE  -  CONT
1  -  LEN
"controller_conveyor_
Send_DB"  -  CONNECT
"SENDBUFFER"  -  DATA
...  -  COM_RST
```

```
TSEND_C
EN
"DONE"  -  DONE
```

```
TSEND_C
EN
"DONE"  -  BUSY
TRUE  -  ERROR
1  -  LEN
"controller_conveyor_
Send_DB"  -  CONNECT
"SENDBUFFER"  -  DATA
...  -  COM_RST
```

```
TSEND_C
EN
"DONE"  -  BUSY
TRUE  -  ERROR
1  -  LEN
"controller_conveyor_
Send_DB"  -  CONNECT
"SENDBUFFER"  -  DATA
...  -  COM_RST
```

```
TSEND_C
EN
"DONE"  -  BUSY
TRUE  -  ERROR
1  -  LEN
"controller_conveyor_
Send_DB"  -  CONNECT
"SENDBUFFER"  -  DATA
...  -  COM_RST
```

```
TSEND_C
EN
"DONE"  -  BUSY
TRUE  -  ERROR
1  -  LEN
"controller_conveyor_
Send_DB"  -  CONNECT
"SENDBUFFER"  -  DATA
...  -  COM_RST
```

```
TSEND_C
EN
"DONE"  -  BUSY
TRUE  -  ERROR
1  -  LEN
"controller_conveyor_
Send_DB"  -  CONNECT
"SENDBUFFER"  -  DATA
...  -  COM_RST
```

```
TSEND_C
EN
"DONE"  -  BUSY
TRUE  -  ERROR
1  -  LEN
"controller_conveyor_
Send_DB"  -  CONNECT
"SENDBUFFER"  -  DATA
...  -  COM_RST
```

```
TSEND_C
EN
"DONE"  -  BUSY
TRUE  -  ERROR
1  -  LEN
"controller_conveyor_
Send_DB"  -  CONNECT
"SENDBUFFER"  -  DATA
...  -  COM_RST
```

```
TSEND_C
EN
"DONE"  -  BUSY
TRUE  -  ERROR
1  -  LEN
"controller_conveyor_
Send_DB"  -  CONNECT
"SENDBUFFER"  -  DATA
...  -  COM_RST
```

```
TSEND_C
EN
"DONE"  -  BUSY
TRUE  -  ERROR
1  -  LEN
"controller_conveyor_
Send_DB"  -  CONNECT
"SENDBUFFER"  -  DATA
...  -  COM_RST
```

```
TSEND_C
EN
"DONE"  -  BUSY
TRUE  -  ERROR
1  -  LEN
"controller_conveyor_
Send_DB"  -  CONNECT
"SENDBUFFER"  -  DATA
...  -  COM_RST
```

```
TSEND_C
EN
"DONE"  -  BUSY
TRUE  -  ERROR
1  -  LEN
"controller_conveyor_
Send_DB"  -  CONNECT
"SENDBUFFER"  -  DATA
...  -  COM_RST
```

```
TSEND_C
EN
"DONE"  -  BUSY
TRUE  -  ERROR
1  -  LEN
"controller_conveyor_
Send_DB"  -  CONNECT
"SENDBUFFER"  -  DATA
...  -  COM_RST
```

```
TSEND_C
EN
"DONE"  -  BUSY
TRUE  -  ERROR
1  -  LEN
"controller_conveyor_
Send_DB"  -  CONNECT
"SENDBUFFER"  -  DATA
...  -  COM_RST
```

```
TSEND_C
EN
"DONE"  -  BUSY
TRUE  -  ERROR
1  -  LEN
"controller_conveyor_
Send_DB"  -  CONNECT
"SENDBUFFER"  -  DATA
...  -  COM_RST
```

```
TSEND_C
EN
"DONE"  -  BUSY
TRUE  -  ERROR
1  -  LEN
"controller_conveyor_
Send_DB"  -  CONNECT
"SENDBUFFER"  -  DATA
...  -  COM_RST
```

```
TSEND_C
EN
"DONE"  -  BUSY
TRUE  -  ERROR
1  -  LEN
"controller_conveyor_
Send_DB"  -  CONNECT
"SENDBUFFER"  -  DATA
...  -  COM_RST
```

```
TSEND_C
EN
"DONE"  -  BUSY
TRUE  -  ERROR
1  -  LEN
"controller_conveyor_
Send_DB"  -  CONNECT
"SENDBUFFER"  -  DATA
...  -  COM_RST
```

```
TSEND_C
EN
"DONE"  -  BUSY
TRUE  -  ERROR
1  -  LEN
"controller_conveyor_
Send_DB"  -  CONNECT
"SENDBUFFER"  -  DATA
...  -  COM_RST
```

```
TSEND_C
EN
"DONE"  -  BUSY
TRUE  -  ERROR
1  -  LEN
"controller_conveyor_
Send_DB"  -  CONNECT
"SENDBUFFER"  -  DATA
...  -  COM_RST
```

```
TSEND_C
EN
"DONE"  -  BUSY
TRUE  -  ERROR
1  -  LEN
"controller_conveyor_
Send_DB"  -  CONNECT
"SENDBUFFER"  -  DATA
...  -  COM_RST
```

```
TSEND_C
EN
"DONE"  -  BUSY
TRUE  -  ERROR
1  -  LEN
"controller_conveyor_
Send_DB"  -  CONNECT
"SENDBUFFER"  -  DATA
...  -  COM_RST
```

```
TSEND_C
EN
"DONE"  -  BUSY
TRUE  -  ERROR
1  -  LEN
"controller_conveyor_
Send_DB"  -  CONNECT
"SENDBUFFER"  -  DATA
...  -  COM_RST
```

```
TSEND_C
EN
"DONE"  -  BUSY
TRUE  -  ERROR
1  -  LEN
"controller_conveyor_
Send_DB"  -  CONNECT
"SENDBUFFER"  -  DATA
...  -  COM_RST
```

```
TSEND_C
EN
"DONE"  -  BUSY
TRUE  -  ERROR
1  -  LEN
"controller_conveyor_
Send_DB"  -  CONNECT
"SENDBUFFER"  -  DATA
...  -  COM_RST
```

```
TSEND_C
EN
"DONE"  -  BUSY
TRUE  -  ERROR
1  -  LEN
"controller_conveyor_
Send_DB"  -  CONNECT
"SENDBUFFER"  -  DATA
...  -  COM_RST
```

```
TSEND_C
EN
"DONE"  -  BUSY
TRUE  -  ERROR
1  -  LEN
"controller_conveyor_
Send_DB"  -  CONNECT
"SENDBUFFER"  -  DATA
...  -  COM_RST
```

```
TSEND_C
EN
"DONE"  -  BUSY
TRUE  -  ERROR
1  -  LEN
"controller_conveyor_
Send_DB"  -  CONNECT
"SENDBUFFER"  -  DATA
...  -  COM_RST
```

```
TSEND_C
EN
"DONE"  -  BUSY
TRUE  -  ERROR
1  -  LEN
"controller_conveyor_
Send_DB"  -  CONNECT
"SENDBUFFER"  -  DATA
...  -  COM_RST
```

```
TSEND_C
EN
"DONE"  -  BUSY
TRUE  -  ERROR
1  -  LEN
"controller_conveyor_
Send_DB"  -  CONNECT
"SENDBUFFER"  -  DATA
...  -  COM_RST
```

```
TSEND_C
EN
"DONE"  -  BUSY
TRUE  -  ERROR
1  -  LEN
"controller_conveyor_
Send_DB"  -  CONNECT
"SENDBUFFER"  -  DATA
...  -  COM_RST
```
Program in ladder diagram (LAD):

Click on “Save project”

In the project navigation window, select “controller_conveyor” and click on the button Download to device.
5.3 Receive Block TRCV_C

Description

TRCV_C is executed asynchronously and has the following functions:

Setting up and establishing a communication connection:

TRCV_C sets up a TCP or ISO-on-TCP communication connection and establishes it. After the connection is set up and established, the CPU maintains and monitors it automatically.
To set up the communication connection, the connection description specified at the parameter CONNECT is used. To establish the connection, the parameter CONT has to be set to "1". If the connection was established successfully, the parameter DONE is set to "1".
When the CPU enters the STOP mode, an existing connection is cancelled and the connection that was set up is removed. To again set up and establish the connection, TRCV_C has to be executed again. The number of possible communication connections is provided in the technical data of your CPU.

Receiving data over an existing communication connection:

If the parameter EN_R is set to "1", data receipt is activated. The data that is received is entered in the receive area. The length of the receive area is specified depending on the protocol variant used; either with the parameter LEN (if LEN <> 0) or the length indication of the parameter DATA (if LEN = 0).
If data receipt was successful, the signal status at parameter DONE is "1". If errors occur during data transmission, the parameter DONE is reset to "0".

Disconnecting the communication connection:

The communication connection is disconnected when the parameter CONT is set to "0".
When setting the parameter COM_RST, TRCV_C is executed again. The existing communication connection is disconnected and a new connection established. Data received while the connection is re-established may be lost.
Receive Modes of TRCV_C

The table below shows how the received data is entered in the receive area.

<table>
<thead>
<tr>
<th>Protocol variant</th>
<th>Availability of data in the receive area</th>
<th>connection_type parameter of the connection description</th>
<th>LEN parameter</th>
<th>RCVD_LEN parameter</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCP (Ad-hoc mode)</td>
<td>The data are immediately available.</td>
<td>E#16#11</td>
<td>65535</td>
<td>1 to 1472</td>
</tr>
<tr>
<td>TCP (Data receipt with specified length)</td>
<td>The data are available as soon as the data length specified at the LEN parameter was fully received.</td>
<td>E#16#11</td>
<td></td>
<td>1 to 8192</td>
</tr>
<tr>
<td>ISO on TCP (protocol-controlled data transfer)</td>
<td>The data are available as soon as the data length specified at the LEN parameter was fully received.</td>
<td>E#16#12</td>
<td></td>
<td>1 to 8192</td>
</tr>
</tbody>
</table>

TCP (ad hoc mode)

The ad hoc mode exists only for the protocol variant TCP. The ad hoc mode is set by assigning the value 0 to the parameter LEN. The length of the receive area is defined by the pointer at the parameter DATA. The data length actually received at the parameter RCVD_LEN has to be identical with the length that was defined at the parameter DATA. A maximum of 8192 bytes can be received.

TCP (data volume with specified length)

The length of data volume is specified with the value of the parameter LEN. The data specified at the parameter DATA is available at the receive area as soon as the length specified at the parameter LEN was received completely.

ISO on TCP (protocol controlled data transmission)

In the case of the protocol variant ISO on TCP, the data is transmitted protocol-controlled. The receiving area is defined by the parameters LEN and DATA.
### Parameters of the TRCV_C

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Declaration</th>
<th>Data type</th>
<th>Memory area</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EN_R</td>
<td>Input</td>
<td>BOOL</td>
<td>I, Q, M, D, L</td>
<td>Receive enable</td>
</tr>
</tbody>
</table>
| CONT         | Input       | BOOL      | I, Q, M, D, L | Controls the communications connection:  
                  • 0: Automatically disconnect communications connection after data have been sent  
                  • 1: Establish and maintain the communications connection  
                  When receiving data (rising edge at the EN_R parameter), the CONT parameter must have the value TRUE in order to establish or maintain a connection |
| LEN          | Input       | UINT      | I, Q, M, D, L or constant | Maximum length of the data to be received. If you use purely symbolic values at the DATA parameter, the LEN parameter must have the value "0". |
| CONNECT      | InOut       | TCON_Param | D           | Pointer to the connection description  
                  See also: Parameters of communication connections |
| DATA         | InOut       | VARIANT   | I, Q, M, D, L | Pointer to the receive area |
| COMM_RST     | InOut       | BOOL      | I, Q, M, D, L | Repeats the instruction:  
                  • 0: Irrelevant  
                  • 1: Complete restart of the instruction causing an existing connection to be terminated |
| DONE         | Output      | BOOL      | I, Q, M, D, L | Status parameter with the following values:  
                  • 0: Job not yet started or is still executing  
                  • 1: Job completed error-free |
| BUSY         | Output      | BOOL      | I, Q, M, D, L | Status parameter with the following values:  
                  • 0: Job not yet started or already completed  
                  • 1: Job not yet completed. A new job cannot be started |
| ERROR        | Output      | BOOL      | I, Q, M, D, L | Status parameter ERROR:  
                  • 0: No error  
                  • 1: Error occurred |
| STATUS       | Output      | WORD      | I, Q, M, D, L | Status of the instruction |
| RCVD_LEN     | Output      | UINT      | I, Q, M, D, L | Amount of data actually received in bytes |
BUSY, DONE and ERROR

The parameters BUSY, DONE, ERROR and STATUS are used to check the execution status. The parameter BUSY shows the processing status. The parameter DONE checks whether a request was executed successfully. The parameter ERROR is set when errors occur while TRCV_C is executed.

<table>
<thead>
<tr>
<th>BUSY</th>
<th>DONE</th>
<th>ERROR</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-</td>
<td>-</td>
<td>The job is being processed.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>The job was completed successfully.</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>The job was stopped with an error. The cause of the error is output at the STATUS parameter.</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>No new job was assigned.</td>
</tr>
</tbody>
</table>

The error information is read out at the parameter STATUS.

<table>
<thead>
<tr>
<th>ERROR</th>
<th>STATUS (00H-1FH)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0009</td>
<td>Job completed error-free</td>
</tr>
<tr>
<td>0</td>
<td>7009</td>
<td>No job processing active</td>
</tr>
</tbody>
</table>
| 0     | 7001             | - Start execution of the job      
'| Establish connection      
| Wait for connection partner |
| 0     | 7002             | Data is being received                                                       |
| 0     | 7003             | Connection is being terminated                                              |
| 0     | 7004             | - Connection established and monitored                                  
'| No job processing active  |
| 1     | 8005             | - The LEN parameter is higher than the highest permitted value.           
'| - The value at the LEN or DATA parameter was changed after the first call  |
<p>| 1     | 8006             | The ID parameter is outside the permitted range.                           |
| 1     | 8007             | Maximum number of connections reached; no additional connection possible    |
| 1     | 8008             | The value at the LEN parameter does not match the receive area set at the DATA parameter |
| 1     | 8009             | The CONNECT parameter does not point to a data block.                      |
| 1     | 8010             | Maximum nesting depth exceeded.                                            |
| 1     | 80A3             | The CONNECT parameter points to a field that does not match the length of the connection description. |
| 1     | 80A4             | The ID of the local device (local_device_id) in the connection description does not match the CPU. |
| 1     | 80A0             | Group error for error codes W#16#80A1 and W#16#80A2.                      |</p>
<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
</table>
| 80A1 | Connection or port already being used by user.  
      | Communications error:  
      | - The specified connection has not yet been established.  
      | - The specified connection is being terminated.  
      | - Transfer over this connection is not possible.  
      | - The interface is being re-initialized. |
| 80A2 | Local or remote port is being used by the system. |
| 80A3 | Attempt being made to re-establish an existing connection.  
      | Attempt being made to terminate a non-existent connection. |
| 80A4 | IP address of the remote endpoint of the connection is invalid, in other words, it matches the IP address of the local partner. |
| 80A7 | Communications error: You called the COM_RST = 1 instruction before the send job was complete. |
| 80B2 | The CONNECT parameter points to a data block that was generated with the attribute Only store in read memory. |
| 80B3 | Inconsistent parameter assignment: Group error for error codes W#16#80A0 to W#16#80A2, W#16#80A4, W#16#80B4 to W#16#80B9. |
| 8184 | "You have violated one or both of the following conditions for passive connection establishment (active_est = FALSE) when using the ISO on TCP protocol variant (connection_type = 0x10#12): "local_tasp_id_len != 8#10#02", and/or "local_tasp_id[1] != 8#10#00"."
| 8185 | Only passive connection establishment is permitted for connection type 13 = UDP. |
| 80B6 | Parameter assignment error in the connection_type parameter of the data block for connection description. |
| 80B7 | Error in one of the following parameters of the data block for connection description: block_length, local_tasp_id_len, rem_subnet_id_len, rem_staddr_len, rem_tasp_id_len, next_staddr_len. |
| 80C3 | All connection resources are in use.  
      | A block with this ID is already being processed in a different priority group. |
| 80C4 | Temporary communications error:  
      | - The connection cannot be established at this time.  
      | - The interface is receiving new parameters or the connection is being established.  
      | - The configured connection is being removed by a "TDISCON" instruction.  
      | - The connection used is being terminated by a call with  COM_RST = 1. |
| 8722 | Error in the CONNECT parameter: Invalid source area (area not declared in data block). |
| 873A | Error in the CONNECT parameter: Access to connection description is not possible (no access to data block). |
| 877F | Error in the CONNECT parameter: Internal fault. |
| 8922 | DATA parameter: Invalid target area, the area does not exist in the DB. |
| 8924 | DATA parameter: Area error in the VARIANT pointer. |
| 8902 | DATA parameter: The DB number is too large. |
| 896A | CONNECT parameter: Access to entered connection data not possible (e.g. because the DB does not exist). |
| 897F | DATA parameter: Internal error, e.g. illegal VARIANT reference. |
| 8A3A | DATA parameter: No access to the data area because, for example, the data block does not exist. |
5.4 Control Program for controller_data

Supplementing the default tag table

Under PLC tags, open the default tag table of the controller_data. Supplement the table.

In OB1 of the control system program, the receive block TRCV_C is called.

Open the Main [OB1] of controller_data
Drag the block "TRCV_C" to Network 1.

In the following window, accept the suggested data block. First click on Single instance and then on "OK".
The data block is generated and incorporated automatically. Under Properties, select the connection partners. First, at the connection data of the local controller, select the existing data block "controller_data_connection_DB", otherwise, a new data block is generated. Then, select the partner controller and the associated data block.
Expand the block to its full size and enter the block parameters either at the block or at the property window.

**TRCV_C**

- **Connection parameter** enabled to receive
  - **EN_R**: TRUE

- **Connection state (CONT)**:
  - 0 = automatic connection termination, 1 = retain connection
  - **CONT**: TRUE

- **Receive length (LEN)**:
  - Maximum number of bytes to receive with the request
  - **LEN**: 1
Generate the **MOVE** command in Network 2.

This completes the programming at the controller_data.

**Program in function block diagram (FBD):**

![FBD Diagram](image)

**Network 1:** call TRCV_C

**Network 2:** copy receive buffer to outputs

![Network Diagrams](image)
Program in ladder diagram (LAD):

1. Click on "Save project".

![Save project]

2. In the project navigation window, select "controller_data" and click on the button **Downloading to device**.

After successful loading to the CPU, the program can be tested.

The input signals I0.0 to I0.7 of the controller_conveyor are displayed at outputs O0.0 to O0.7 of the controller_data.